Assignment 2: Performance evaluation of different architectures on QEMU

Department of Electrical, Computer and Software Engineering
SOFE 4590U: Embedded Systems(Fall 2023)
Professor Akramul Azim

Prepared by:

Name	Student ID
Jean-Paul Saliba	100741759

Introduction:

The main goal of this assignment is to utilize the QEMU emulator to analyze and compare the performance of object detection and image processing for two different architectures. The two architectures that have been utilized, include Debian x86 architecture and the i386 architecture. The iso file of both architectures can be retrieved online through the QEMU website, which is required before running the commands by running any commands in the command prompt.

GitHub Code Link: https://github.com/Jp-s24/Embedded-Assignment-2

Video Demo Link: In the Github and within this drive link:

https://drive.google.com/drive/folders/1CE_GU4Rtkas9md4-x21rl1z7kuA03hkw?usp=sharing

Description of Architectures Used:

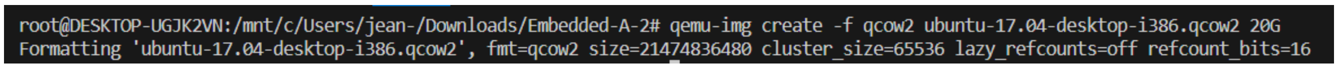
i386 Ubuntu Architecture:

A. Setup

1. Creating the Image:

The command below demonstrates the initial step, which is the creation of the image for the i386 architecture. This command creates an image cluster and allocates 20G to that image created.

```
qemu-img create -f qcow2 ubuntu-17.04-desktop-i386.qcow2 20G
```

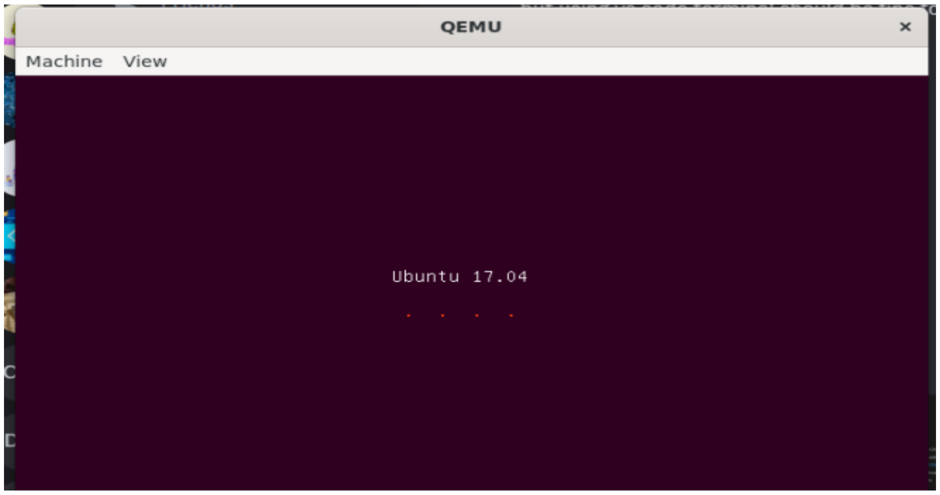


```
root@DESKTOP-UGJK2VN:/mnt/c/Users/jean-/Downloads/Embedded-A-2# qemu-img create -f qcow2 ubuntu-17.04-desktop-i386.qcow2 20G
Formatting 'ubuntu-17.04-desktop-i386.qcow2', fmt=qcow2 size=21474836480 cluster_size=65536 lazy_refcounts=off refcount_bits=16
```

2. Booting up the Operating System with the use of the iso file:

This command below starts the architecture using the iso image while opening the Ubuntu OS.

```
qemu-system-i386 -m 2G -boot d -cdrom ubuntu-17.04-desktop-i386.iso
```



B. Running the Image Processing and Object Detection Program:

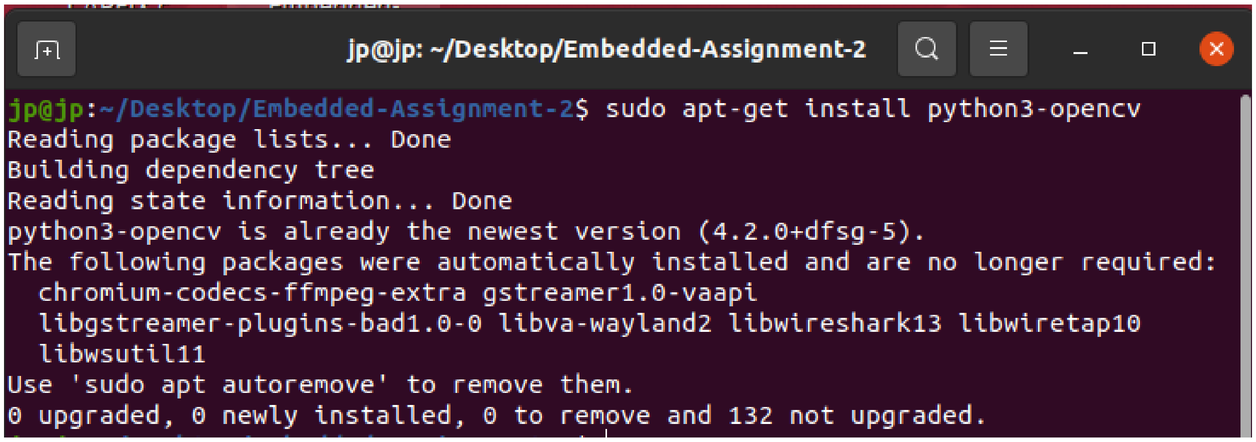
1. Clone the Python program and before running the program ensure all files were cloned correctly.

- The command below was run to clone the GitHub repository.

```
git clone https://github.com/Jp-s24/Embedded-Assignment-2.git
```

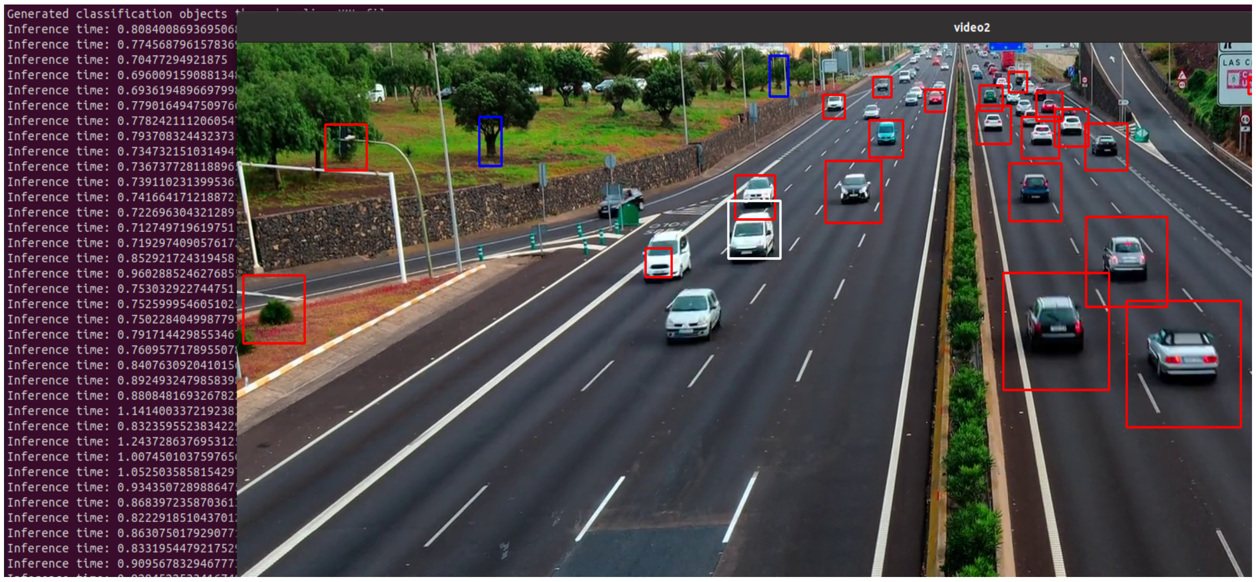
2. Run the Python program:

Note: Before running the Python object detection program you must install the opencv python package first.



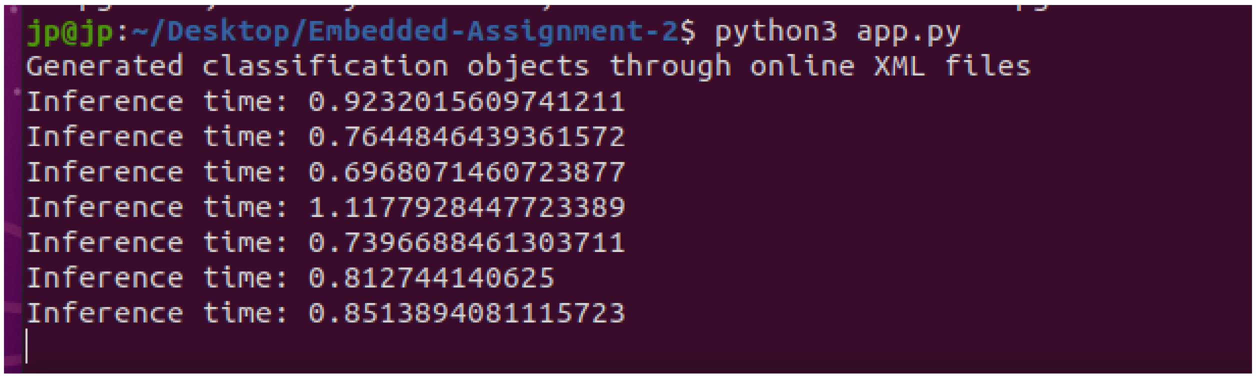
```
jp@jp:~/Desktop/Embedded-Assignment-2$ sudo apt-get install python3-opencv
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-opencv is already the newest version (4.2.0+dfsg-5).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi
  libgstreamer-plugins-bad1.0-0 libva-wayland2 libwireshark13 libwiretap10
  libwsutil11
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 132 not upgraded.
```

After you run the app.py file via the python3 app.py command the output is as follows:



```
Generated classification objects t
Inference time: 0.80840086936950
Inference time: 0.77456879615783
Inference time: 0.70477294921875
Inference time: 0.69600915908813
Inference time: 0.69361948966979
Inference time: 0.77901649475097
Inference time: 0.77824211120605
Inference time: 0.793708324432373
Inference time: 0.73473215103149
Inference time: 0.73673772811889
Inference time: 0.73911023139953
Inference time: 0.74166417121887
Inference time: 0.72269630432128
Inference time: 0.712749719619751
Inference time: 0.71929740905761
Inference time: 0.852921724319458
Inference time: 0.96028852462768
Inference time: 0.753032922744751
Inference time: 0.75259995460510
Inference time: 0.75022840499877
Inference time: 0.79171442985534
Inference time: 0.76095771789550
Inference time: 0.84076309204101
Inference time: 0.89249324798583
Inference time: 0.88084816932678
Inference time: 1.14140033721923
Inference time: 0.83235955238342
Inference time: 1.24372863769531
Inference time: 1.00745010375976
Inference time: 1.05250358581542
Inference time: 0.93435072898864
Inference time: 0.86839723587036
Inference time: 0.82229185104370
Inference time: 0.86307501792907
Inference time: 0.83319544792175
Inference time: 0.90956783294677
```

C. Results:



```
jp@jp:~/Desktop/Embedded-Assignment-2$ python3 app.py
Generated classification objects through online XML files
Inference time: 0.9232015609741211
Inference time: 0.7644846439361572
Inference time: 0.6968071460723877
Inference time: 1.1177928447723389
Inference time: 0.7396688461303711
Inference time: 0.812744140625
Inference time: 0.8513894081115723
```

X86 Debian Architecture:

A. Setup

1. The command below is the initial step where the image is created

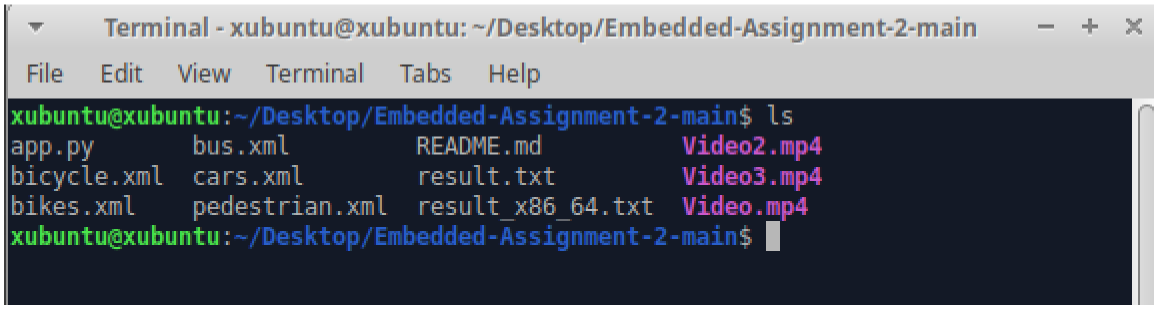
```
qemu-img create -f qcow2 Xubuntu.img 20G
```

2. Start the architecture after downloading the iso file for the x86 architecture. The command below can be run in the command prompt and will boot up the x86 architecture.

```
qemu-system-x86_64 -drive file=xubuntu.qcow2 -cdrom xubuntu.iso -boot
d -m 4096 -net nic -net user -accel whpx
```

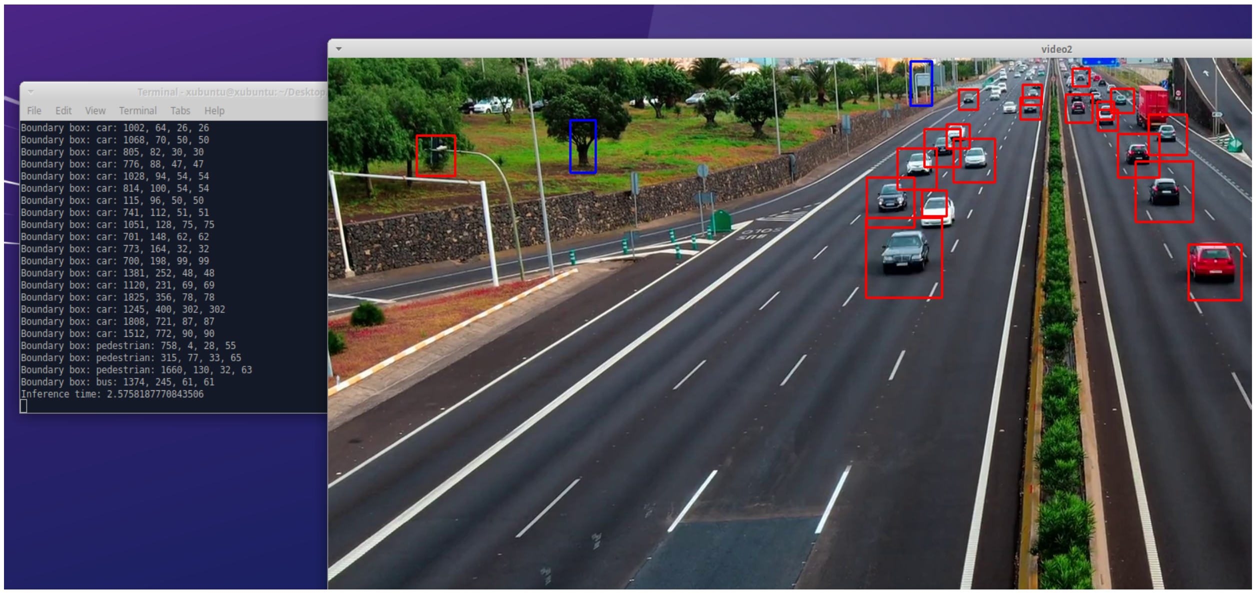
B. Running the Program

1. Ensure all files are on the PC and the code is organized in one folder.


Terminal - xubuntu@xubuntu: ~/Desktop/Embedded-Assignment-2-main
File Edit View Terminal Tabs Help

```
xubuntu@xubuntu:~/Desktop/Embedded-Assignment-2-main$ ls
app.py        bus.xml         README.md          Video2.mp4
bicycle.xml   cars.xml        result.txt         Video3.mp4
bikes.xml     pedestrian.xml  result_x86_64.txt  Video.mp4
xubuntu@xubuntu:~/Desktop/Embedded-Assignment-2-main$
```

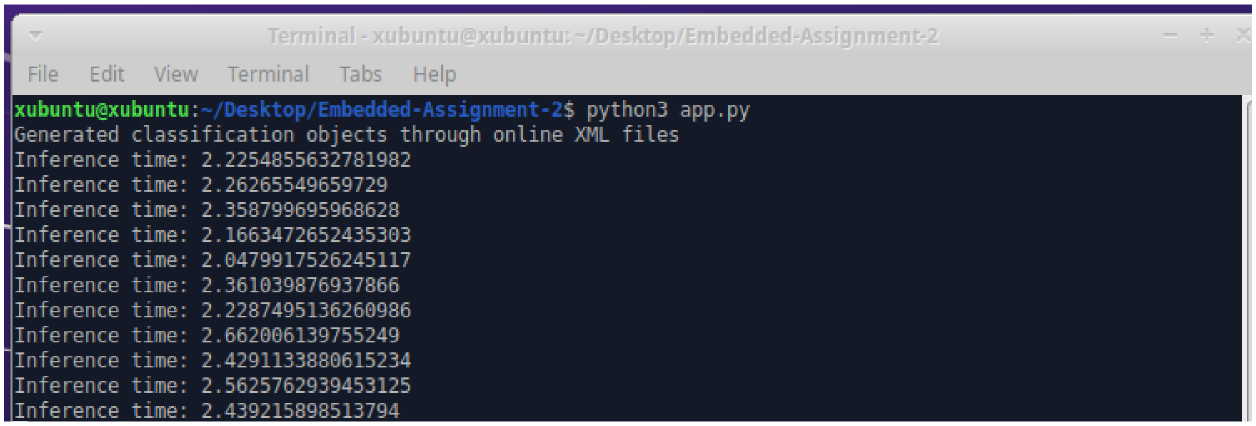
2. Run the program


Terminal - xubuntu@xubuntu: ~/Desktop
File Edit View Terminal Tabs Help
video2

```
Boundary box: car: 1002, 64, 26, 26
Boundary box: car: 1068, 70, 50, 50
Boundary box: car: 805, 82, 30, 30
Boundary box: car: 776, 88, 47, 47
Boundary box: car: 1028, 94, 54, 54
Boundary box: car: 814, 100, 54, 54
Boundary box: car: 115, 96, 50, 50
Boundary box: car: 741, 112, 51, 51
Boundary box: car: 1051, 128, 75, 75
Boundary box: car: 701, 148, 62, 62
Boundary box: car: 773, 164, 32, 32
Boundary box: car: 700, 198, 99, 99
Boundary box: car: 1381, 252, 48, 48
Boundary box: car: 1120, 231, 69, 69
Boundary box: car: 1825, 356, 78, 78
Boundary box: car: 1245, 400, 302, 302
Boundary box: car: 1808, 721, 87, 87
Boundary box: car: 1512, 772, 90, 90
Boundary box: pedestrian: 758, 4, 28, 55
Boundary box: pedestrian: 315, 77, 33, 65
Boundary box: pedestrian: 1660, 130, 32, 63
Boundary box: bus: 1374, 245, 61, 61
Inference time: 2.5758187770843506
```

3. Results

The results below demonstrate the app.py image processing program output for the x86 architecture. The screenshot shows the interference times for various objects detected.


Terminal - xubuntu@xubuntu: ~/Desktop/Embedded-Assignment-2
File Edit View Terminal Tabs Help

```
xubuntu@xubuntu:~/Desktop/Embedded-Assignment-2$ python3 app.py
Generated classification objects through online XML files
Inference time: 2.2254855632781982
Inference time: 2.26265549659729
Inference time: 2.358799695968628
Inference time: 2.1663472652435303
Inference time: 2.0479917526245117
Inference time: 2.361039876937866
Inference time: 2.2287495136260986
Inference time: 2.662006139755249
Inference time: 2.4291133880615234
Inference time: 2.5625762939453125
Inference time: 2.439215898513794
```

Comparison of Two Architectures For Object Detection:

The object identification and image processing program were executed by both the i386 Ubuntu architecture and the x86 Debian architecture and the following conclusions were made. The selection between the two architectures might depend upon particular project needs, resource accessibility, and aspects like interoperability. Based on the results after executing the object detection program in both architectures it was clear that the x86 architecture was less efficient than the i386 in terms of object detection. This is demonstrated through analyzing the interference time of both architectures because the overall interference time is larger for x86 architecture compared to i386 architecture.

References:

Code: https://github.com/search?q=QEMU%20object%20detection&type=repositories

Videos: https://pixabay.com/videos/search/cars%2c%20bikes/